Deep Submicron Backdoors

Ortega, Alfredo
aortega@groundworkstech.com

Buenos Aires, Argentina

Abstract

Malicious hardware is a mature topic but previous research has focused almost exclusively on theoretical applications. In this article, practical implementations of gate-level backdoors will be presented using the Verilog hardware description language, then simulated and finally synthesized using freely available deep sub-micron (45-180 nm) standard cells, resulting in a backdoored latest-generation ARM CPU, suitable for fabrication and massive deployment.

Contents

1 Revision history

The following table shows the revision history for this document.

Date	Version	Revision
03/28/2014	FINL20140401	First version.

2 Introduction

Cyberattacks are shaping the modern warfare, targeting software instead of physical objects. Software and firmware attacks are becoming more and more relevant given their low price and high effectiveness. Naturally as attacks increase, new software and firmware defenses are being created and deployed. The bar for a effective software-only attack is much higher now than 5-years before. That's why hardware-only attacks become attractive. Tools for hardware-level auditing are non-existent or immature at best, making it an attractive target for current attacks. We can expect than as software and firmware security mature, hardware attacks will become more and more common, even if they may not be as practical or cost-effective as a software-only attack. There are many examples of successful hardware attacks in the wild over the years. A modern computer or mobile phone is composed of multiple hardware sub-systems, all of which can be compromised. There are many kinds of attacks. A rough taxonomy may be:

Debugging interfaces: Those are backdoors unintentionally left by the vendor used in the development stage. It's often too expensive to remove them from the final product, and they may present a vulnerability if found and used by a malicious attacker.

Weakening of security-related hardware: A malicious vendor could intentionally weaken a random number generator [14], reduce strength or bypass an encryption algorithm, or leak credentials through side-channels.

Malicious logic implants: This consist in additional circuitry that once activated allows the attacker to control or extract data from the target system via various methods, breaking the security model design of the system or directly modifying the software or firmware of the target system.

In this article we will describe an attack of the third type, a malicious logic implant, creating a backdoor in a commercial CPU, an ARM cortex M0. The backdoor is inserted at the Verilog level, and then it's synthesized using different processes to analyze the impact that the additional logic may have.

Figure 1: Replica of the Great Seal which contained a Soviet bugging device, on display at the NSA's National Cryptologic Museum. (Source: Wikipedia)

3 Previous works

Previous work in hardware-level attacks are too many to cite. For historical purposes one of the more famous hardware attacks was the Thing listening device (See Fig. 1). This hardware backdoor or "Implant" used passive techniques to transmit an audio signal, and it was energized and activated by electromagnetic energy from an outside source. Designed by Leon Theremin, it was covertly located in the the US embassy in Moscow until it was exposed in 1952.

The clipper chip [13] was a chipset developed by the U.S. National Security Agency (NSA) as an encryption device for voice transmission. It used a data encryption algorithm called SkipJack. This implemented the concept of key escrow. From Wikipedia:

> In the factory, any new telephone or other device with a Clipper chip would be given a "cryptographic key", that would then be provided to the government in "escrow". If government agencies "established their authority" to listen to a communication, then the key would be given to those government agencies, who could then decrypt all data transmitted by that particular telephone. The newly formed Electronic Frontier Foundation preferred the term "key surrender" to emphasize what they alleged

was really occurring.

This was not only unacceptable from a privacy point of view, but in 1994, Matt Blaze published an article [1] that pointed out that the Clipper's escrow system had a serious cryptographic vulnerability. The chip was no longer relevant in 1996 due to lack of adoption.

3.1 NSA ANT Catalog

In 2014, one of the documents leaked by Edward Snowed revealed a 50 page classified document listing technology available to the NSA Tailored Access Operations (TAO) by the ANT division to aid in cyber surveillance. This catalog-like document list hundreds of products that any ANT division agent can order for tapping their targets' data. This includes software and hardware devices. Hardware backdoors available in the catalog range from remote USB-taps to remotely illuminated active tempest[9]-like surveillance, with a price from free (for software implants) to US$250,000.

3.2 Non-gov examples

Some hardware backdoors are created by private companies and have valid and legal uses in determinate situations. We can cite Anti-theft technology [2], where a combination of software and hardware some PC chipsets alerts the legitimate owner of the equipment when it has been stolen. In this case, the backdoor must be stealthy and persistent to defeat the thief. Some backdoors are actually required by law, for example Lawful Intercept Modules are sometimes required in network equipment, they allow for the Law enforcement agencies to intercept and listen private conversations between suspected criminals. This backdoor-like systems often add a layer of complexity with their own vulnerabilities [5] and always present the risk of abuse. Academic study of malicious hardware is also a very active field, for example the IEEE Symposium on Hardware-Oriented Security and Trust (HOST) and the NYU-Poly Embedded-System Challenge are annual events where experts meet to discuss potential hardware attacks and protections.

3.3 Unintentional backdoors

Debugging interfaces are a clear example of an unintentional backdoor. Modern digital computer systems often ship with powerful development aids, in

the form of debugging interfaces like the JTAG Port [10]. Developers often add their own set of higher-level debugging aids like root shells, development credentials, etc. Functionally these tools are indistinguishable from backdoors and often are identified as such. A malicious attacker can take advantage of those interfaces and easily subvert the attacked system. It is important to remove this extensions before the final version of the system, but often they are forgotten and most of the time, they are not documented. A possible example of an unintentional hardware backdoor is the a particular functionality present in the PROASIC3 FPGA, identified by Skorobogatov Et. Al. [11].

4 Backdoor Implementation

We chose to implement the backdoor as a Verilog module that attaches to the bus and takes control of the system memory when certain conditions trigger its activation. This kind of implant is not stealth but it's very practical and easy to implement and study. As a target CPU we chose the ARM Cortex M0, as its source code is easily accessible by research and educational institutions at low or no cost. More powerful open-source CPUs exist, but we believe that implementation on a commercial platform is important to demonstrate the feasibility of this kind of attacks. Another reason for choosing the ARM Cortex M0 is that the design is ASIC-ready, meaning that unlike most open-source soft-cpus [1], it efficiently synthesizes in FPGAs and ASICs. This allowed us to implement and simulate the backdoor in both an Altera FPGA and two ASIC processes. It is important to note that the final step of the ASIC process, the physical fabrication, was not completed at this stage of the research.

5 MALPROXY: BUS malware

The backdoor that we call MALPROXY was implemented as a small malicious state machine that constantly monitors the bus for a 56-bit activation word and attached command. For a detailed description of the command structure, see 5. The attacker must craft the malicious command and cause the CPU to place it in the bus, for example sending it through the network

[1] Soft-cpu: CPU built entirely as a FPGA state machine

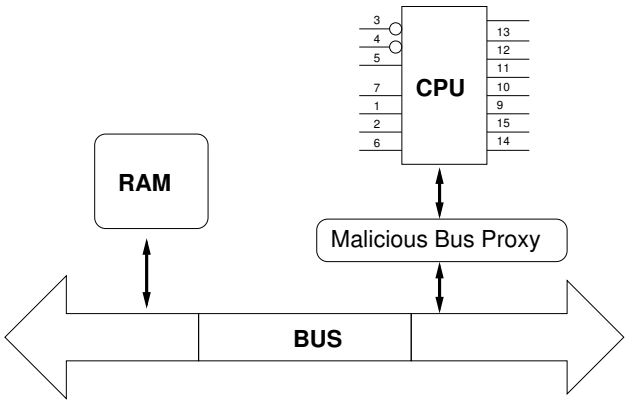

Figure 2: High-level design of MALPROXY Bus backdoor

as ICMP Ping payload or embedding it in a file and causing the CPU to store the file into memory. Any of those actions will force the malicious command to be transferred via the AMBA bus and to be exposed to the monitoring circuit in the backdoor. The backdoor subsequently activates and executes the command execution (See state-machine description at Fig. 3). The whole command can be hundreds of bits long and it's often broken into many non-consecutive segments, so a basic defragmentation and assembly of the command is required in the backdoor logic.

The following Verilog code and ascii schematic shows the malicious command format. It uses 3 different "magic numbers" or cookies to signal that the data is a part of a command. The whole command is 160 bits wide, with multiple cookies inside to facilitate reassembly:

```
`define RTK_COOKIE_1 32'h12345678
`define RTK_COOKIE_2 24'h434241
`define RTK_COOKIE_3 24'h2D2D2D
`define RTK_CMD_WRITE  "W"
`define RTK_CMD_READ  "R"

          <------32 bits ------->
       -> [RTK_COOKIE_1]
       -> [RTK_COOKIE_2 + Command]
       -> [RTK_COOKIE_3 + DATA]*4
       -> [RTK_COOKIE_3 + ADDR]*4
       -> [RTK_COOKIE_3 + EXEC] : Executes Command
```

The backdoor is compatible with the Advanced Microcontroller Bus Architecture or AMBA, the standard ARM bus and the de-facto industry standard embedded and system-on-chip bus, used in many applications like smart phones. This allows it to be compatible with all CPUs that support the AMBA bus, and specifically supports the AMBA-lite bus, the version that the ARM Cortex M0 uses. It can be said this is not a CPU backdoor but more appropriately a BUS backdoor, as no CPU logic is modified. However, the physical location after the synthesis, place and route is inside the CPU die. We expected that this would make it harder to detect as both the gates and interconnections of the backdoor and the CPU are intimately mixed. Later it was found that this estimation was incorrect, and that differences in code style allows to easily locate different logic visually.

5.1 High-level design

As can be seen in Fig. 2, the backdoor is logically located between the bus and the CPU, allowing it to monitor all incoming and outgoing data and addresses. There should be a small negligible delay caused by the bus signal passing through an additional multiplexer. When a correct command is detected and parsed, the backdoor briefly disconnects the CPU form the BUS. This disconnection lasts about a BUS cycle, or 2 clocks. With the control of the BUS, the backdoor can modify memory or access any peripheral. However only two commands are necessary to take control of the system, a read_memory primitive and a write_memory primitive. If the memory layout

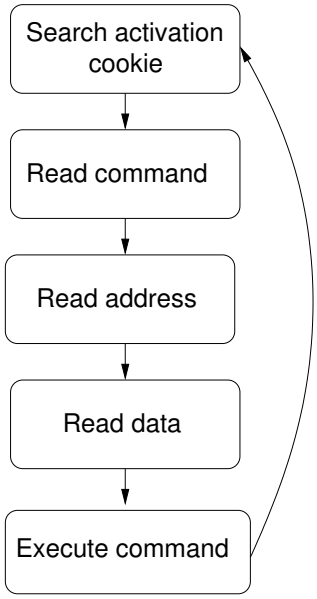

Figure 3: Implanted state machine

is known by the attacker and variations are small or no software protections are present, often only the write_memory primitive is enough to take control of the system by inserting malicious software as a second stage attack and redirecting the execution flow by, for example, modifying the contents of a function pointer.

5.2 implementation

The ARM Cortex M0 verilog code provided by the Designstart program [6] is provided in three files:

cortexm0ds_logic.v: contains the (obfuscated) main logic.

CORTEXM0DS.v: Macro cell wrapper.

armtb.v: Testbench. Contains simulations of a terminal, clock, RAM and a small AMBA-lite implementation.

Currently the backdoor is implemented inside the macro cell wrapper as a 100-line Verilog module. The armtb.v file contains all peripheral needed for successful simulation.

5.3 Simulation and software

Verilog simulation can be done using the popular open-source tool Icarus Verilo [7]. To compile the simulator, issue this command line:

```
~$ iverilog CORTEXM0DS.v cortexm0ds_logic.v armtb.v
```

And the hardware simulation is ready to be executed. But even if the backdoor is complete and operational, the system needs to be running software that activates it. We can use any software compatible with the Cortex M0. We chose the ChibiOS/RT free embedded RTOS [8], slightly modified to make it compatible with our simulation hardware. Makefiles and source code are provided in the project site.

The main thread is a simple function consisting in only 9 C lines:

```
1  int main(void) {
2    char buf[40];
3    char *str="\x78\x56\x34\x12WABCA---A---A---A
4               ---\x00---\x00---\x0a---\x65---";
5    while (TRUE) {
6      puts("Main thread: hello world");
7      memcpy(buf,str,40);
8      chSchDoRescheduleBehind();
9    }
10 }
```

This specific line:

```
  char *str="\x78\x56\x34\x12WABCA---A---A---A
             ---\x00---\x00---\x0a---\x65---";
```

Contains the hardware backdoor activation word and an encoded write_memory operation that writes the 32-bit string "AAAA" into a specific memory position where the operative system stores string constants. This command is activated after it is being placed in the BUS via the apparently innocuous memcpy() operation.

After compiling the software and running the simulation, the system starts outputting text via the simulated terminal. Even if the package Icarus

Verilog is a interpreted simulator, the design is small so it have acceptable simulation performance. ChibiOS initializes in less than 5 seconds:

```
~$ ./a.out
                    0: ------------------------------------------------
                    0: ARM(r) Cortex(tm)-M0 DesignStart(tm) Testbench
                    0: (c) Copyright 2010 ARM Limited
                    0: All Rights Reserved
                    0: ------------------------------------------------
                    0: Backdoor by Groundworks Technologies
                    0: http://www.groundworkstech.com
                    0: ------------------------------------------------
                    0: Loading initial memory content...
                    0: Loading completed
Main thread: hello world
Main AAAAad: hello world
Main AAAAad: hello world
```

We can see how the first printed line is the original, but after the memcpy() operation, the content of the printed string changes, and the "AAAA" string inserted by the MALPROXY backdoor appears.

5.4 Physical syntesis

For the complete design flow, Cadence's SOC Encounter software package was used. There are many steps from the Hardware Description Language (HDL) source to the GDSII file suitable for fabrication, with multiple verification and simulations to assure the validity of the design, as a mistake or miscalculation in the design can cost millions of dollars. For the sake of simplicity, many of those steps were skipped, and only the basic Logic synthesis, place and route steps are detailed. Also many important steps needed in a real design like power rings and external connections or gate fillings, are also skipped.

The first step, logical synthesis, compiles the Verilog files into a netlist consisting in logical gates and their connections. The type, area and amount of gates will depend of the process technology used. For example, 180nm gates are bigger than 45 nm gates, but also the 45nm process may allow optimized gates not present at all in the 180nm process technology. The

type and layout of gates available in a process are specified in a library file called "Standard Cell Library" usually provided by the vendor.

Many free Standard Cell Libraries are available. Virgina Tech university offers a free 350nm, 250nm and 180nm standard cell library targeting the same TSMC process technology [3]. For example, the 180nm vtvt_tsmc180 library contains 83 standard cells, including multiple input AND, OR, NAND, NOR, XNOR, etc., adders and flip-flops optimized for 180nm. This library is suitable for fabrication, for example using the Mosis service [**?**], that can be free of charge for educational institutions. As the 180nm process technology is ancient by today standards, it's also very low cost.

The NanGate corporation offers a modern 45nm Open-source cell library suitable for testing and exploring EDA (Electronic Design Automation) flows, but not for fabrication as it does not target any real 45nm process technology. This library includes 170 different standard cells.

The choice of process technology not only determines the amount and type of standard cells used, but also the metal layers, or interconnections among gates. For example, the TSMC 180nm allows for 6 metal layers (and one polysilicon layer to form the gate contacts), but the Nangate 45nm has a total of 10 metal layers and one poly, allowing more integration and more complex interconnections.

This is a small TCL script with the commands that can be used to synthesize the system using Encounter design compiler. Note that no clock is specified so timing is always met:

```
# Nangate 45nm libs.
# For TSMC 180nm use ‘‘set_attribute library {osu018_stdcells.lib}’’
set_attribute library {NangateOpenCellLibrary_typical_ccs.lib}

## This read the files. Main ARM logic:
read_hdl cortexm0ds_logic.v

## Macro cell block and Backdoor:
read_hdl -v2001 CORTEXM0DS.v

## This buils the general block
elaborate

##This synthesizes your code
synthesize -to_mapped

## This writes all your files
write -mapped > CORTEXM0DS_synth.v
```

The output of this step is also a Verilog file, but with additional information about logical cells. Fig. 4 shows a graphical representation of the logical design output. After this step, place , route and some automated design rule checks can be done using the Encounter GUI. Fig. 5 shows the top view of the physical synthesis.

5.5 Implementation results

The commands “Report area” and “Report power” can be used inside the RTL Compiler utility as a rought estimation of power and area occupied. The following table contains the results of all process technology tried.

Parameter	**180nm TSMC**	**45nm Nangate**
Total cells	9084	6418
Total area	0.38 mm^2	0.046 mm^2
Total power	9.8 (mW)	0.78 (mW)
Backdoor cells	379	230
Backdoor area	0.01 mm^2	0.003 mm^2
Backdoor power	0.22 (mW)	0.059 (mW)

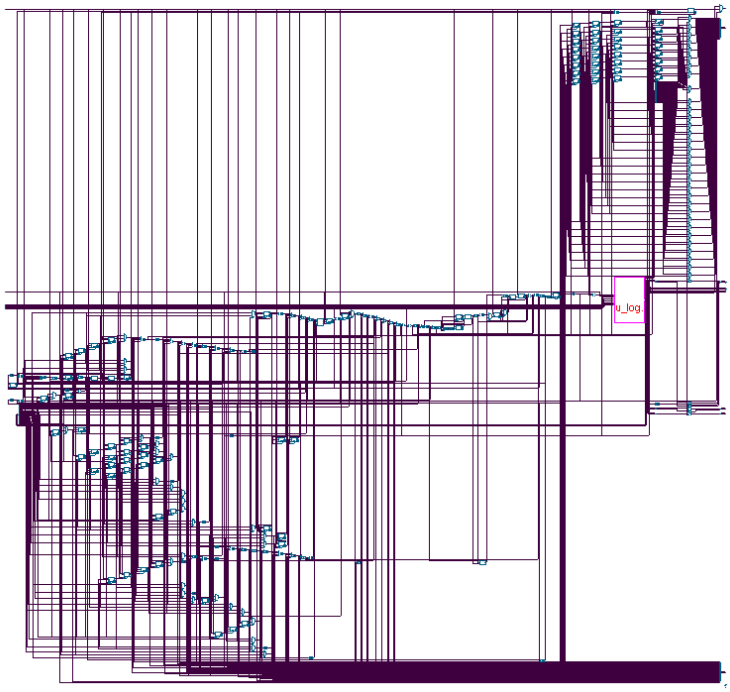

Figure 4: Result of the logic synthesis. MALPROXY backdoor logic gates are blue. ARM CPU block is marked as "u_log" in the red box.

We can see that the backdoor presence increase CPU power and area resources by about 5%. Taking in account that the Cortex M0 is the smallest available ARM CPU, increases in resource usage for normal ARM or bigger CPUs should be negligible.

6 Contermeasures

Auditing of the design by reverse-engineering and comparing with known-good designs seems to be the first step facing a suspected backdoored ASIC. No tools for aiding this process are known to the authors of this article at this time. However interest in the area is increasing and some works [15] on decapping [2] and IC reverse-engineering are being published, the first

[2] Decaping consist in using nitric acid to remove the top and study the chip die layer by layer

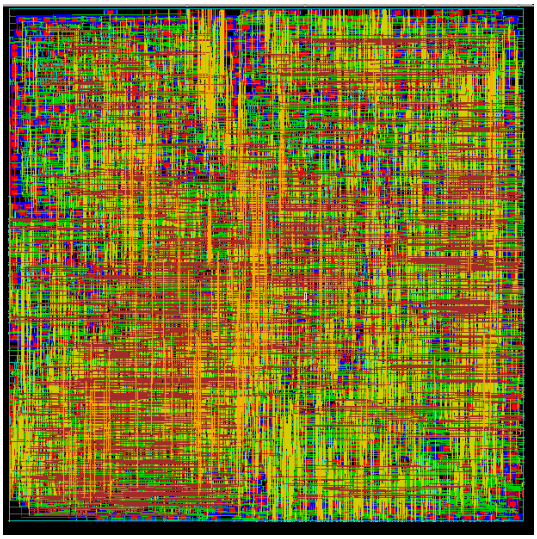

Figure 5: Top view of the physical synthesis result, 180nm TSMC Tech.

steps needed for auditing capability of VLSI ASICs. Complexity of the task facing suspected big designs, however, may frustrate or make impossible any auditing effort with bigger CPUs or ASICs.

7 Conclusions

In this article we enumerated the steps to implement a hardware backdoor in a commercial ARM CPU. The final result is actually an AMBA-lite bus backdoor implemented inside the CPU die. Simulations proved that a the backdoor can be made to work and be activated with bus traffic caused by a memcpy or similar operation, as an example of bus transfers hidding commands to parallel logic. An interesting result of targeting the bus instead of the CPU is that the backdoor can be made relatively CPU-independent, as long as the bus parameters and size do not change. Another finding is that code-style differences can make the malicious logic evident because of the different amount of routing layers used in different modules. However this may not be true for bigger designs as the Cortex M0 is a very small CPU.

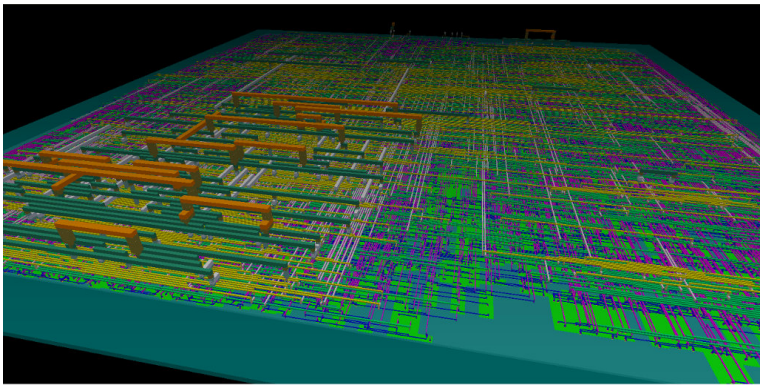

Figure 6: 3D view of the physical synthesis result using GDS3D [12], 45nm NanGate Tech. Note routing differences in the malicious logic area vs regular ARM Cortex M0 area.

References

[1] "Protocol Failure in the Escrowed Encryption Standard.", *http://www.analog.com/en/processors-dsp/blackfin/vdsp-bf-sh-ts/products/product.html*, Proceedings of Second ACM Conference on Computer and Communications Security.

[2] Intel anti-theft technology, *http://www.intel.com/content/www/us/en/architecture-and-technology/anti-theft/anti-theft-general-technology.html*, Intel corporation.

[3] Virginia Tech, *Cell Libraries to Support VLSI Research and Education*, url: `http://www.vtvt.ece.vt.edu/vlsidesign/cell.php`

[4] *MOSIS Integrated Circuit Fabrication Service*, url: `http://www.mosis.com`

[5] Ortega, Alfredo, and Anibal Sacco. *"Deactivate the Rootkit: Attacks on BIOS anti-theft technologies."*, Blackhat USA 2009.

[6] DesignStart for Processor IP, *http://www.arm.com/products/processors/designstart-processor-ip/index.php*, ARM corporation

[7] Williams Stephen, *"Icarus verilog* ***http://iverilog.icarus.com****"*, Open-Source verilog simulator, 2006.

[8] ChibiOS/RT free embedded RTOS, ***http://www.chibios.org***

[9] Young, J. , *"NSA tempest documents."*, ***http://cryptome.info/0001/nsa-tempest***, 2008-06-141.

[10] Whetsel, Lee, "A View of the JTAG Port and Architecture.", ATE & Instrumentation Conference West. 1988.

[11] Skorobogatov, Sergei, and Christopher Woods, *"Breakthrough silicon scanning discovers backdoor in military chip"*, (05 March 2012)

[12] Velner, Soer, ***http://sourceforge.net/projects/gds3d/***, University of Twente.

[13] Levy, Steven, *"Battle of the Clipper chip."*, New York Times Magazine, Sunday (1994): 44.

[14] Schneier, Bruce, *"Did NSA Put a Secret Backdoor in New Encryption Standard?."*, URL: `http://www.wired.com/politics/security/commentary/securitymatters/2007/11/securitymatters_1115`, (2007-11-15) (2007).

[15] Alex Radocea, *"Literacy for Integrated Circuit Reverse Engineering"*, Ekoparty Security Conference, 2012.